Formal Reasoning for Explainable Neural Network Classification

AiChemist Summer School

Michael Wand

michael.wand@supsi.ch

Dalle Molle Institute for Artificial Intelligence (IDSIA) USI - SUPSI

16.6.2026

Introduction

- In this tutorial we present Formal Reasoning for Explainable AI (XAI), specifically in the context of Neural Networks (NN).
- This method offers major advantages over "classical" approaches, in particular:
  - ➔ guarantees on the behavior of NNs which can be expressed in mathematical terms
  - ➔ a powerful algorithmic approach to create versatile explanations with diverse properties.

Introduction

- In this tutorial we present Formal Reasoning for Explainable AI (XAI), specifically in the context of Neural Networks (NN).
- This method offers major advantages over "classical" approaches, in particular:
  - ➔ guarantees on the behavior of NNs which can be expressed in mathematical terms
  - ➔ a powerful algorithmic approach to create versatile explanations with diverse properties.
- We will cover the following topics:
  - ➔ Basics of Formal Reasoning
  - ➔ Elements of Classical XAI
  - ➔ Formal Explanations of Neural Network Classification

Team

- We will cover some of our own work (Space Explanations).
- Collaboration between IDSIA and the Formal Verification and Security Lab of USI (Università della Svizzera italiana).
- The Team:

Natasha Sharygina[1]

Tomáš Kolárik[1]

Grigory Fedyukovich[1,3]

Faezeh Labbaf[1]

Fabrizio Leopardi[1]

Michael Wand[2]

[1] University of Lugano (USI), Switzerland
[2] SUPSI, IDSIA, Switzerland
[3] Florida State University, USA

This study was partially funded by Hasler Foundation, Switzerland, as part of the project "Formal Reasoning on Neural Networks"

idsia

Basics of Formal Reasoning

Basics

- Formal Reasoning: draw conclusions using precise symbolic rules.
- Rules are encoded in a Formal System.
- Goal: enable automated reasoning
- This is the basis of important fields like program verification, theorem proving, automated planning, etc.

Image courtesy ChatGPT

Propositional Logic

- Propositional Logic: the most basic formal system.
- Building blocks: Variables (true or false) and Connectives.
- Basic connectives:
  - ➔ Conjunction $P \wedge Q$
  - ➔ Disjunction $P \vee Q$
  - ➔ Negation: $\neg P$

Propositional Logic

- Propositional Logic: the most basic formal system.
- Building blocks: Variables (true or false) and Connectives.
- Basic connectives and their semantic interpretations:
  - ➔ Conjunction $P \wedge Q$ true if both P and Q are true
  - ➔ Disjunction $P \vee Q$ true if at least one of P and Q are true
  - ➔ Negation: $\neg P$ true if P is false, and vice versa

Propositional Logic

- Propositional Logic: the most basic formal system.
- Building blocks: Variables (true or false) and Connectives.
- Basic connectives and their semantic interpretations:
  - ➔ Conjunction $P \wedge Q$ true if both P and Q are true
  - ➔ Disjunction $P \vee Q$ true if at least one of P and Q are true
  - ➔ Negation: $\neg P$ true if P is false, and vice versa
- Propositional formulas can be manipulated using the rules of Boolean Algebra.
- Example:

$$P \wedge (Q \vee R) \Leftrightarrow (P \wedge Q) \vee (P \wedge R).$$

Propositional Logic

- Propositional Logic: the most basic formal system.
- Building blocks: Variables (true or false) and Connectives.
- Basic connectives and their semantic interpretations:
  - ➔ Conjunction $P \wedge Q$ true if both P and Q are true
  - ➔ Disjunction $P \vee Q$ true if at least one of P and Q are true
  - ➔ Negation: $\neg P$ true if P is false, and vice versa
- Propositional formulas can be manipulated using the rules of Boolean Algebra.
- Example:

$$P \wedge (Q \vee R) \Leftrightarrow (P \wedge Q) \vee (P \wedge R).$$

- By giving a specific interpretation to the variables, propositional formulas acquire meaning.
- Example: $R =$ it is raining, $O =$ I am outside, $D =$ I am dry. Now,

$$R \wedge O \implies \neg D.$$

Propositional Logic

- A formula is interpreted by assigning truth values to the variables.
- Example: if $P = \text{true}, Q = \text{false}$, then

$$(P \vee Q) \wedge \neg Q \qquad \text{is true}$$
$$(Q \vee \neg P) \wedge (P \wedge Q) \qquad \text{is false.}$$

- A formula is satisfiable is it is true for some assignment of truth values to the variables.
- Example: the formula

$$(P \vee Q) \wedge (\neg Q \vee R) \wedge (P \vee \neg R)$$

  is satisfiable by setting $P = \text{true}$ and $Q = \text{false}$.
- The formula

$$(A \vee B) \wedge (\neg A \wedge \neg B)$$

  is unsatisfiable (no assignment of the variables makes it true).

Satisfiability

- The Boolean Satisfiability Problem (SAT) is the task to determine whether a given formula is satisfiable (in this case we are interested in a specific example of variable assignments) or not.
- One of the famous classical problems in Computer Science, it is known to be NP-complete.

Satisfiability

- The Boolean Satisfiability Problem (SAT) is the task to determine whether a given formula is satisfiable (in this case we are interested in a specific example of variable assignments) or not.
- One of the famous classical problems in Computer Science, it is known to be NP-complete.
- This means that we have no chance of finding an algorithm which solves this problem efficiently in the *general* case, as soon as there are more than a few variables.
- In other words, in the general case, no algorithm is better than trying out all (exponentially many) combinations of variable assignments until the formula evaluates as true, or all possibilities are exhausted and no satisfying assignment has been found.

Satisfiability

- The Boolean Satisfiability Problem (SAT) is the task to determine whether a given formula is satisfiable (in this case we are interested in a specific example of variable assignments) or not.
- One of the famous classical problems in Computer Science, it is known to be NP-complete.
- This means that we have no chance of finding an algorithm which solves this problem efficiently in the *general* case, as soon as there are more than a few variables.
- In other words, in the general case, no algorithm is better than trying out all (exponentially many) combinations of variable assignments until the formula evaluates as true, or all possibilities are exhausted and no satisfying assignment has been found.
- end of this talk????

SAT Solvers

- Despite the exponential worst-case complexity of SAT, many algorithms have been presented (and implemented) to solve *typical* cases efficiently!

SAT Solvers

- Despite the exponential worst-case complexity of SAT, many algorithms have been presented (and implemented) to solve *typical* cases efficiently!
- Starting point: a formula in Conjunctive Normal Form (CNF).
  - ➔ Each formula is an AND of clauses.
  - ➔ Each clause is an OR of literals.
  - ➔ Example:

$$(P \vee Q \vee \neg S) \wedge (\neg Q \vee R) \wedge (\neg T) \qquad \text{is in CNF}$$
$$(A \vee B) \wedge (\neg A \vee (\neg B \wedge C)) \qquad \text{is not in CNF.}$$

  - ➔ Any formula can be converted to an *equisatisfiable* CNF formula in polynomial time...
  - ➔ ...and as we are going to see, in some cases we already start with a formula in CNF.

SAT Solvers

- Despite the exponential worst-case complexity of SAT, many algorithms have been presented (and implemented) to solve *typical* cases efficiently!
- Starting point: a formula in Conjunctive Normal Form (CNF).
  - ➔ Each formula is an AND of clauses.
  - ➔ Each clause is an OR of literals.
  - ➔ Example:

$$(P \vee Q \vee \neg S) \wedge (\neg Q \vee R) \wedge (\neg T) \qquad \text{is in CNF}$$
$$(A \vee B) \wedge (\neg A \vee (\neg B \wedge C)) \qquad \text{is not in CNF.}$$

  - ➔ Any formula can be converted to an *equisatisfiable* CNF formula in polynomial time...
  - ➔ ...and as we are going to see, in some cases we already start with a formula in CNF.
- Modern SAT solvers can deal with formulas with millions of variables and tens of millions of clauses.

SAT Solvers: DPLL

- Classical algorithm: DPLL (Davis–Putnam–Logemann–Loveland, 1961).
- Basic idea: Recursion and backtracking
  1. Choose a variable and assign a value (true or false)
  2. Simplify formula (in particular, remove all clauses which now evaluate as true)
  3. Recurse: go to step 1 and choose another variable
  4. If an assignment which evaluates the formula as true is found, return it, otherwise go to step 1 with *the same* variable, but the *opposing* truth value
  5. if no assignment is found even with the opposing truth value, the formula is unsatisfiable.
- Clauses with only one element (unit clauses) immediately yield a single possible assignment for the concerned variable.
- Since variables are recursively eliminated, unit clauses can appear frequently, which greatly reduces the search complexity.

SAT Solvers: CDCL

- Modern flavor of SAT solvers: CDCL (Conflict Driven Clause Learning, 1996 - now)
- Key innovations:
  - ➔ Conflict detection: when a conflict is detected (e.g. a clause becomes entirely false), the solver analyzes why it happened and *adds* a new clause preventing this mistake.
  - ➔ For example, if the solver learns: $A = \text{true}, B = \text{true}$ implies a conflict, then it derives the clause $\neg A \vee \neg B$.
  - ➔ Backjumping: The history of decisions is kept as a graph; when a conflict is detected, the source of the conflict is detected, and the algorithm restarts where the conflict occurred.
- The algorithm works well in practice because typical problems do not have a random structure: they exhibit repeated patterns and symmetries which are exploited by the learning part of the algorithm.

Satisfiability Modulo Theories (SMT)

- The classical SAT problem deals with boolean variables, which may be true or false.
- One can endow these boolean variables with meaning by replacing them with complex statements.
- The statements are formally expressed in a theory.
- We are especially interested in the theory of linear real arithmetic (LRA), which allows to express statements like

$$x_1 + 2x_2 - 3.4x_3 \leq 10$$
$$-8.5x_2 + x_4 = 23.$$

Satisfiability Modulo Theories (SMT)

- Comparison of a standard boolean statement and an SMT statement with linear real arithmetic:

SAT	SMT
$(P \vee \neg Q \vee R) \wedge$	$[(x_1 + 3x_2 - 6.7x_4 \leq 10) \vee (x_1 - 4x_3 \geq 23)] \wedge$
$(Q \vee \neg S) \wedge$	$[(3.4x_1 - 17x_4 = -15) \vee (x_2 < x_3)] \wedge$
$(\neg P \vee \neg R \vee S)$	$[(-2x_2 + 3x_3 - 4x_4 > 9.3)]$

- Clearly, this is an even harder problem than basic SAT.
- Yet, in the case of LRA an exact solution can often be found in reasonable time (or it can be shown that the problem is unsatisfiable).
- Note that these algorithms are sound: they will yield a result which is provably mathematically correct, *not* an approximation.

SMT Solving

- Assume a statement in conjunctive normal form.
- The general flow of SMT solvers comprises two components:
  - ➔ The SAT Engine handles the boolean skeleton of the formula.
  - ➔ The Theory Solver takes a set of clauses and attempts to satisfy them simultaneously.

SMT Solving

- Assume a statement in conjunctive normal form.
- The general flow of SMT solvers comprises two components:
  - ➔ The SAT Engine handles the boolean skeleton of the formula.
    - ➔ Each clause is represented by a boolean variable.
    - ➔ The SAT Engine suggests a truth assignment of these clauses.
    - ➔ The set of clauses which must be simultaneously true is passed to the Theory Solver.
  - ➔ The Theory Solver takes a set of clauses and attempts to satisfy them simultaneously.
    - ➔ In the case of linear arithmetic, usually a variant of the simplex algorithm is used.

SMT Solving

- Assume a statement in conjunctive normal form.
- The general flow of SMT solvers comprises two components:
  - ➔ The SAT Engine handles the boolean skeleton of the formula.
    - ➔ Each clause is represented by a boolean variable.
    - ➔ The SAT Engine suggests a truth assignment of these clauses.
    - ➔ The set of clauses which must be simultaneously true is passed to the Theory Solver.
  - ➔ The Theory Solver takes a set of clauses and attempts to satisfy them simultaneously.
    - ➔ In the case of linear arithmetic, usually a variant of the simplex algorithm is used.
- In practical implementations, there is a continuous feedback loop between these two components.
- For example, if the theory solver finds a contradiction (e.g. $x_2 < 2 \wedge x_2 > 5$), a suitable clause to avoid this conflict can be added to the original formula, in the spirit of CDCL.

The Simplex Algorithm

- The Theory Solver usually uses a variant of the Simplex Algorithm (Dual Simplex, Feasability Simplex).
- In contrast to the classical form, here the goal is not to optimize a variable, but to check whether a feasible solution exists at all.
- The technique is similar: equations are held in a *tableau*, when a contradiction is found the algorithm tries to *pivot* variables, etc.

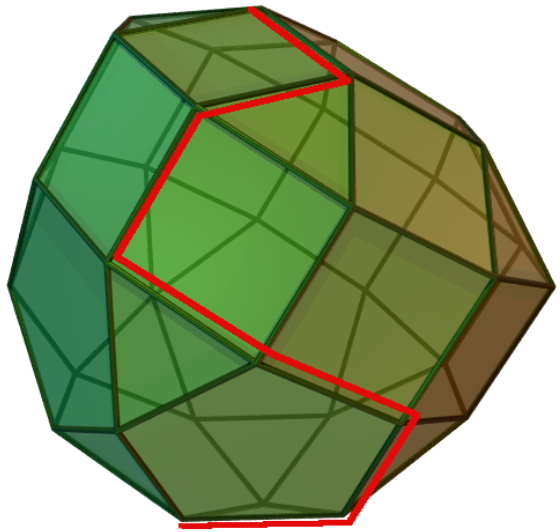


Image: Wikipedia, Simplex Algorithm

- Very important component: incrementality (when constraints are added or removed by the SAT engine, the intermediate status of the Simplex engine is kept).
- To avoid rounding errors, SMT(LRA) uses exact rational arithmetic (i.e. values are saved as fractions), not floating point numbers.

idsia

Elements of Classical XAI

Elements of Explainable AI

- The field of Explainable AI (XAI) has become very extensive and diverse in the past years, so I will *not* attempt to give a complete taxonomy.
- We will instead consider a few well-known methods, to get an idea about typical strengths, weaknesses, and open questions . . .

Elements of Explainable AI

- The field of Explainable AI (XAI) has become very extensive and diverse in the past years, so I will *not* attempt to give a complete taxonomy.
- We will instead consider a few well-known methods, to get an idea about typical strengths, weaknesses, and open questions . . .
- . . . which formal reasoning can tackle.

Elements of Explainable AI

idsia

- The field of Explainable AI (XAI) has become very extensive and diverse in the past years, so I will *not* attempt to give a complete taxonomy.
- We will instead consider a few well-known methods, to get an idea about typical strengths, weaknesses, and open questions . . .
- . . . which formal reasoning can tackle.
- Some basic definitions:
  - ➔ We are interested in post-hoc explanations of trained neural networks (i.e. we do not change the NN architecture to facilitate explanation).
  - ➔ We only cover classifying neural networks.
  - ➔ Local methods explain why a specific sample is classified in a certain way, Global methods explain the overall logic of the model across the entire feature space.
- We are mostly interested in local methods.

The "Classical" Methods (1)

- Saliency methods[1] compute the gradient of the output w.r.t. each input feature (for example, each pixel).
  - ➔ This amounts to a linear approximation of the NN around the given input sample.
- Layer-Wise Relevance Propagation (LRP)[2] is a principled way to distribute "relevance" across the layers of a network.
  - ➔ In the most simple case, relevance is based on the weights of the NN, again yielding a linear approximation.

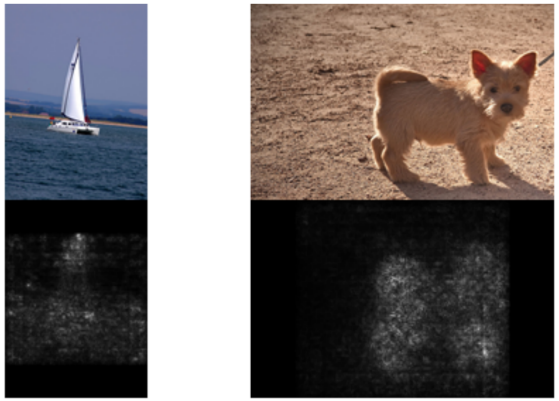


Image: Simonyan et al.

[1] Simonyan et al, *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*. Workshop Proceedings of the ICLR, 2014

[2] Bach et al, *On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation*. PLOS One, 2015

The "Classical" Methods (2)

- **LIME**[3] computes an "interpretable" model locally around a sample point.
  - ➔ For example, a linear model.
- **SHAP (ley values)**[4] are computed by stochastically measuring a feature's average contribution to the model prediction.
  - ➔ Theoretical idea: train the model with each feature present or absent, compare results.
  - ➔ In practice, there are different ways to estimate what a model would predict if features were missing.
  - ➔ Still, exponential complexity.

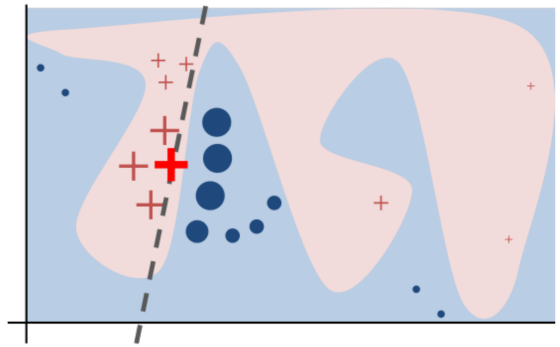


Image: Ribeiro et al.

[3] Ribeiro et al: *"Why Should I Trust You?": Explaining the Predictions of Any Classifier*. Proc. SIGKDD 2016

[4] Lundberg et al: *A Unified Approach to Interpreting Model Predictions*. Proc. NIPS 2017

Criticism of Classical Methods

- **Discussion:** What are possible drawbacks of the "classical" methods presented so far?

Criticism of Classical Methods

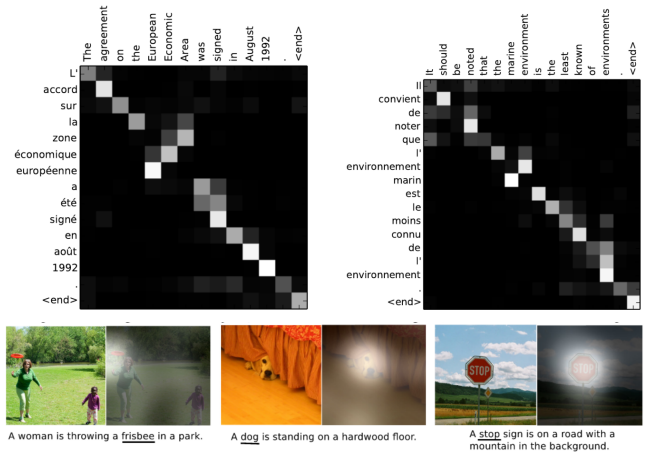
- **Discussion:** What are possible drawbacks of the "classical" methods presented so far?
  - ➔ Frequently based on linearization around a sample
  - ➔ Only valid in an infinitesimally small neighborhood of the sample
  - ➔ A reasonable approximation, but no actual guarantees

Criticism of Classical Methods

- **Discussion:** What are possible drawbacks of the "classical" methods presented so far?
  - ➔ Frequently based on linearization around a sample
  - ➔ Only valid in an infinitesimally small neighborhood of the sample
  - ➔ A reasonable approximation, but no actual guarantees
- Which steps can be taken to make more theoretically/mathematically founded statements about the computational process of NNs?

Attention-based XAI

- Attention-based XAI arises naturally from NN Attention methods.
- The idea is to take the attention weights as an indicator which information is processed during inference.
- Visually striking results have been derived using this technique.
- Still, the method has also been criticized[5].



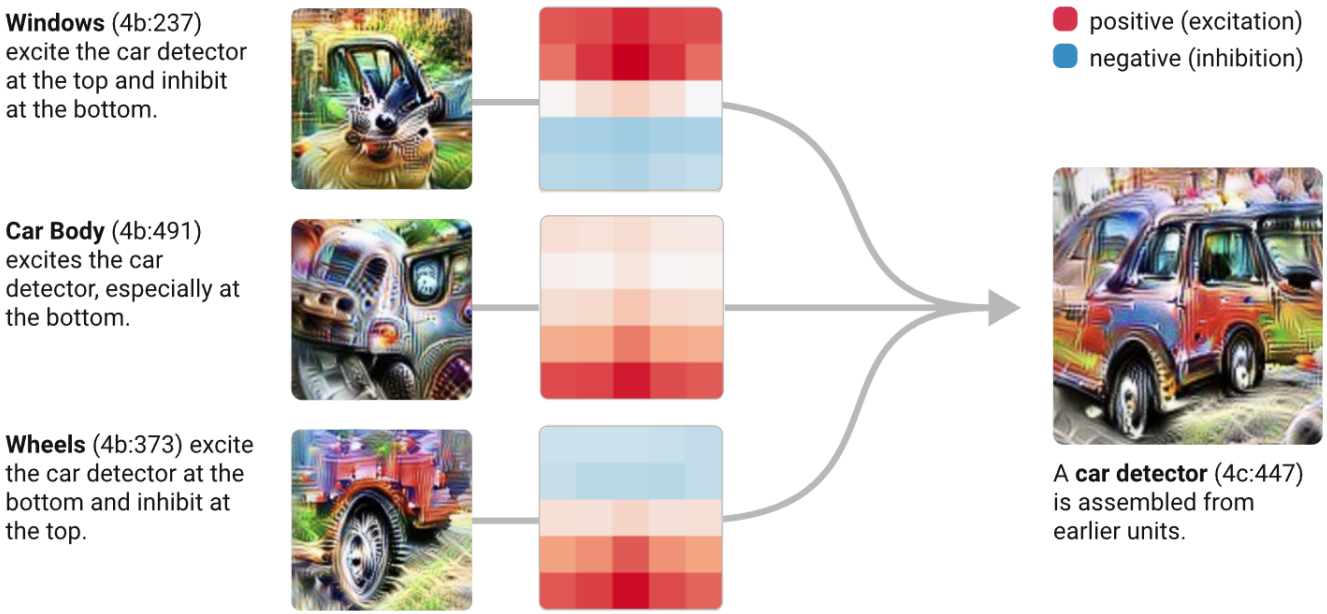
Images: Bahdanau et al, Xu et al.

[5] Jain et al, *Attention is not Explanation*. Proc. NAACL 2019

Mechanistic XAI

- Mechanistic XAI aims to reverse engineer the computational process of a given neural network.
- Attempts to answer the question: *What computation inside the NN produced the prediction?*
- Methods: tracing information flow, analysis of neural representations, ...
- Possibly the currently fastest-growing field of Explainable AI.
- Open issues include scalability and the *distributedness* of representations

Mechanistic XAI



Highly recommended reading: `https://distill.pub/2020/circuits`, `https://transformer-circuits.pub`; image is from distill.pub

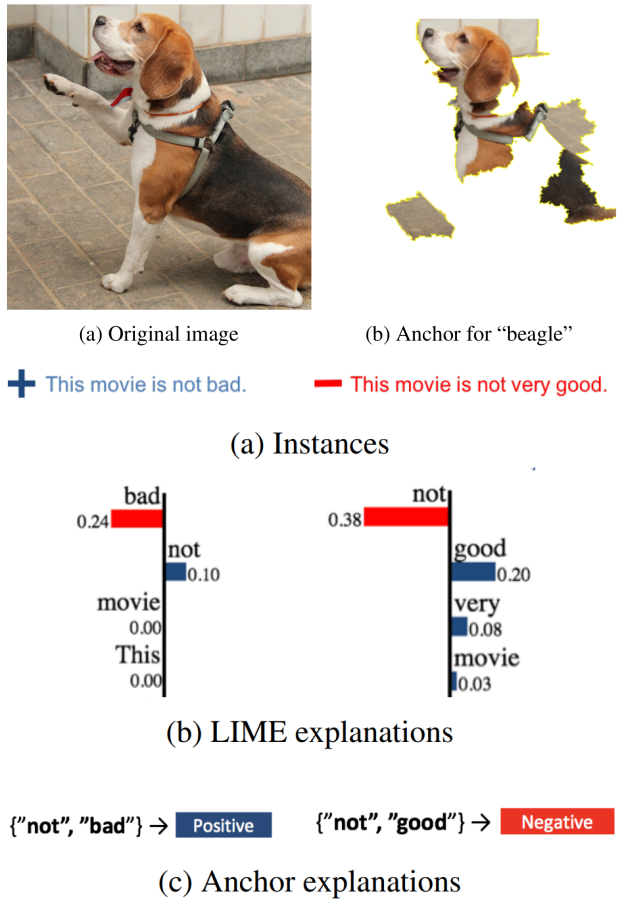
Causal XAI

- Causal Explainable AI aims at identifying which factors were *causally* responsible for a model's decision: A step towards theoretical grounding of XAI statements[6].
- In particular, distinguish cause and correlation!
- Based on the Halpern–Pearl theory of actual causality: A feature is an explanation if changing it would change the outcome under appropriate counterfactual interventions.
  - ➔ Model the prediction as a causal system
  - ➔ Generate counterfactual worlds by intervening on features.
  - ➔ Determine whether a feature (or set of features) is an actual cause of the prediction.
- Example: Loan application rejected. Feature attribution: Income contributed 35%, debt 25%. Causal explanation: Had income been above 60k, the application would have been approved.
- Existing work still uses stochastic approximation.

[6] Chockler/Halpern: *Responsibility and blame: a structural-model approach*. Journal of Artifical Intelligence Research, 2004.

Anchors

- The Anchors method[7]: another step towards making more precise statements in Explaninable AI.
- Idea: Identify a subset of features which guarantees a certain classification with high probability.
- Model-agnostic, sampling/perturbation based method.



[7] Ribeiro et al: *Anchors: High-precision model-agnostic explanations*. Proc. AAAI, 2018

Anchors

- Definition (simplified): Let f be a classifier (for example a NN). Let x be a sample and A be a subset of its features; let $\mathcal{D}(z \mid A)$ be a conditional distribution of perturbations around the fixed features A.
- Then A is an anchor if

$$\mathbb{E}_{\mathcal{D}(z|A)}\left[\mathbf{1}_{f(x)=f(z)}\right] \geq \tau$$

  that is, if these perturbations do not change the classification with high probability.
- Expectation is approximated by sampling.
- Drawback: Different Anchors (subsets of features) are possible.

idsia

Formal Explanations of Neural Network Classification

Abductive Explanations

- Abductive Explanation[8]: for a given sample, find the smallest set (subset-minimal or cardinality-minimal) of features which constitutes a sufficient condition for the given classification.
- This means that all other features can be chosen *arbitrarily* (realistically, within finite bounds).

[8] Ignatiev et al: *Abduction-Based Explanations for Machine Learning Models*. Proc. AAAI, 2019.

Encoding a Neural Network

- Major contribution: formulate the task of finding abductive explanations in a logical framework!
- This allows to use existing solvers (for example SMT/LRA solvers) for finding explanations.
- Fundamentally different from the Anchors methods, we obtain *precise* and *provably correct* statements!

Encoding a Neural Network

- Major contribution: formulate the task of finding abductive explanations in a logical framework!
- This allows to use existing solvers (for example SMT/LRA solvers) for finding explanations.
- Fundamentally different from the Anchors methods, we obtain *precise* and *provably correct* statements!
- Caveat: Neural Networks are highly *nonlinear*, whereas LRA only allows to express statements from *linear* algebra!
- Fortunately, many modern NNs are based on the ReLU activation function, which can be expressed with linear constraints!

Encoding a Neural Network

- Encode the ReLU nonlinearity by using indicator variables z_i.
- The z_i indicate whether a neuron is active (positive input $\Rightarrow$ positive output), or inactive (negative input $\Rightarrow$ zero output).

- The following example encodes a single layer with two input and two outputs.
- Note the slack variables s_i (typical for the simplex algorithm and its variants).
- These conditions must be simultaneously true, yielding a formula ϕ_{NN} in Conjunctive Normal Form.

Example 2 *Consider an example of a block with two inputs, x_1 and x_2 and two outputs y_1 and y_2. Let $A = [2, -1; 1, 1]$ and $b = [-1, 1]$ be parameters of a linear transformation. To encode this block, we introduce auxiliary variables s_1, s_2, z_1 and z_2. We obtain the following constraints:*

$$
\begin{aligned}
2x_1 - x_2 - 1 &= y_1 - s_1, \\
x_1 + x_2 + 1 &= y_2 - s_2, \\
z_1 = 1 \rightarrow y_1 \leq 0, z_2 = 1 \rightarrow y_2 \leq 0, \\
z_1 = 0 \rightarrow s_1 \leq 0, z_2 = 0 \rightarrow s_2 \leq 0, \\
y_1 \geq 0, y_2 \geq 0, s_1 \geq 0, s_2 \geq 0, z_1 \in \{0,1\}, z_2 \in \{0,1\}.
\end{aligned}
$$

Source: Ignatiev et al.

- Now use this idea to encode the entire neural network, layer by layer.

Finding Abductive Explanations

- Assume that the neural network is encoded as given on the previous slide, yielding a formula ϕ_{NN}.
- An input sample $\mathbf{s}$ is given by an assignment of values s_i to the features x_i:

$$(x_1 = s_1 \wedge x_2 = s_2 \wedge \ldots \wedge x_N = s_N) =: \phi_{\text{feat}}.$$

- The classification output is likewise encoded as

$$\bigwedge_{i \neq j} y_i^{(L)} > y_j^{(L)} =: \phi_{\text{class}},$$

where the $y_i^{(L)}$ stand for the neurons of the last layer, and i is the hypothesized class.

Finding Abductive Explanations

- Clearly,

$$\phi_{\text{feat}} \wedge \phi_{\text{NN}} \Rightarrow \phi_{\text{class}}.$$

- Now the algorithm proceeds by iteratively removing constraints from the formula ϕ_{feat}, at each step checking whether the implication remains valid.
- For example, let

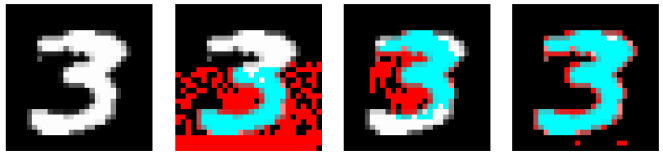
$$\phi_{\text{feat2,3,7}} = (x_2 = s_2 \wedge x_3 = s_3 \wedge x_7 = s_7).$$

If $\phi_{\text{feat2,3,7}} \wedge \phi_{\text{NN}} \Rightarrow \phi_{\text{class}}$, then features 2, 3, and 7 are an abductive explanation for the full sample $\mathbf{s} = (s_1, s_2, \ldots, s_N)$.

- The explanation is subset-minimal if removing any of the constraints breaks the implication, it is cardinality-minimal if no set of two features (or less) satisfies the implication.

Finding Abductive Explanations

- The figure shows some possible subset-minimal explanations for the MNIST digit “3”. What do you think of these explanations?



(a) (b) (c) (d)

Figure 2: Possible minimal explanations for digit three.

Figure: Ignatiev et al

Verix

- Verix: one of the most important follow-ups in applying formal logic to explainability[9].
- Verix specifically targets deep neural networks.
- Assume an input sample is classified as class c. Question: Which regions/features are provably sufficient for the classifier to keep predicting c, *even under perturbations*?
- Perturbations are considered on a per-feature basis.

[9] Wu et al: *VeriX: Towards Verified Explainability of Deep Neural Networks*. Proc. NeurIPS 2023

- Definition: Given a neural network f, an input x, a manipulation magnitude ε, and a discrepancy δ, a robust explanation with respect to norm $p \in \{1, 2, \infty\}$ is a set of input features x_A such that if $B = \bar{A}$, then

$$\forall x'^B : \quad \|x'^B - x^B\|_p \leq \varepsilon \implies |f(x) - f(x')| \leq \delta$$

  where x' is the input variant combining the fixed features x^A and the perturbed features $x^{B'}$.
- The features x^B are called irrelevant features.
- Intuitively, if the features x^A are fixed, the irrelevant features do not influence the prediction...
- ...as long as they do not vary too much.
- A robust explanation is optimal if the set A is minimal.
- This means that if a feature in A is perturbed, one can find a *counterfactual example* where the classification actually changes.
- Relevant idea in the context of *Adversarial Examples*.

Verix

- Verix explanations are computed feature by feature.
- Start: Initialize irrelevant (B) and explanation (A) features as empty lists
- For all features χ_i:
  - ➔ Formulate the precondition that χ_i *and* all irrelevant features found so far can be perturbed by ε.
  - ➔ Check if this precondition still implies the postcondition that the sample classification does not change.
  - ➔ If the implication is true, add χ_i to the list of irrelevant features - otherwise it is part of the explanation.
- The order of the features can be chosen based on some heuristic metric.

Algorithm 1 VERIX (VERIfied eXplainability)

Input: neural network f and input $\mathbf{x} = \langle \chi^1, \ldots, \chi^d \rangle$
Parameter: ϵ-perturbation, norm p, and discrepancy δ
Output: optimal robust explanation $\mathbf{x}^{\mathbf{A}}$, counterfactuals $\mathbf{C}$ for each $\chi \in \mathbf{x}^{\mathbf{A}}$

function VERIX($f, \mathbf{x}$)
  $\mathbf{A}, \mathbf{B}, \mathbf{C} \mapsto \emptyset, \emptyset, \emptyset$
  $c \mapsto f(\mathbf{x})$
  $\hat{c} \mapsto f(\hat{\mathbf{x}})$
  $\pi \mapsto$ TRAVERSALORDER($\mathbf{x}$)
  for i in π **do**
    $\mathbf{B}^+ \mapsto \mathbf{B} \cup \{i\}$
    $\phi \mapsto (\left\| \hat{\chi}^{\mathbf{B}^+} - \chi^{\mathbf{B}^+} \right\|_p \leq \epsilon)$
    $\phi \mapsto \phi \wedge (\hat{\chi}^{\Theta \setminus \mathbf{B}^+} = \chi^{\Theta \setminus \mathbf{B}^+})$
    $(\texttt{HOLD}, m) \mapsto$ CHECK($f, \phi \Rightarrow |\hat{c} - c| \leq \delta$)
    if HOLD **then** $\mathbf{B} \mapsto \mathbf{B}^+$
    else $\mathbf{A} \mapsto \mathbf{A} \cup \{i\}$; $\mathbf{C} \mapsto \mathbf{C} \cup \{m\}$
  return ($\mathbf{x}^{\mathbf{A}}$, $\mathbf{C}$)

Verix

- Technical drawbacks of Verix:
  - ➔ The resulting explanations depend very heavily on the order of feature traversal.
  - ➔ No principled way to choose ε - and what happens when a feature is perturbed by more than ε?
  - ➔ Due to the above, distinction into irrelevant features and explanation features is somewhat artificial.
- There is further related work (Verix+) which aims at providing a more systematic way to order features and obtain interesting explanations, but the fundamental problem remains.

Inflated Explanations

- The concept has been generalized by Inflated Explanations[10].
- Idea: instead of distinguishing irrelevant and explanation features, consider *all* features and compute the range in which they can vary for the classification to hold.
- This yields the following definition: Let $\mathcal{F}$ be the index set of features, and $\mathbb{F}$ be the entire feature space. An inflated explanation of a given sample $\mathbf{v} = (v_j)_{j \in \mathcal{F}}$, which is classified as class c ($\kappa(\mathbf{v}) = c$) is a set of intervals $\mathbb{E}_j, j \in \mathcal{F}$ such that

$$\forall (x \in \mathbb{F}) : \left[\bigwedge_{j \in \mathcal{F}} (x_j \in \mathbb{E}_j) \right] \rightarrow (\kappa(x) = c)$$

and the $\mathbb{E}_j$ are "maximal sets" such that this condition holds.

[10] Izza et al: *Delivering Inflated Explanations*. Proc. AAAI 2024

Inflated Explanations

- As for Verix:
  - ➔ features are traversed in any suitable order
  - ➔ the intervals $\mathbb{E}_j$ are derived by performing linear or exponential search, at each step checking whether the condition from the previous slide holds
  - ➔ the result depends on the order of traversal.
- Still, the explanation is more balanced:
  - ➔ each feature might allow some perturbation without changing the classification result
  - ➔ clearly, some features may be perturbed quite a lot (then they are little relevant for the classification), some features must remain close to the values given by the sample $\mathbf{v}$ (these can be considered as highly relevant).
- Very time-consuming calculation! (Even more than for Verix.)

Discussion: Formal versus Classical XAI

Discussion: What are properties of inflated explanations? How do they compare to classical methods (e.g. Saliency methods)?

Space Explanations

- Space Explanations[11] aim at resolving the fundamental problem of all formal explanations methods presented so far: *Features are considered one-by-one, yielding no information about relationships between them.*
  - ➔ Example 1: assume a medical task (e.g. risk prediction). A subject with high age and high blood pressure might be considered at risk, a younger subject with lower blood pressure might be classified as low-risk patient. But how are critical blood pressure and age related?
  - ➔ Example 2: consider an abductive explanation in image classification (e.g. from Verix, slide 37). The subset of pixels which forms the classification appears to be rather random - maybe instead of forcing a few pixels to have specific values, we should consider average values, or differences, for a group of pixels?
- We will see that advanced methods in formal logic allow to derive such versatile constraints.

[11] Labbaf et al: *Space Explanations of Neural Network Classification*. Proc. CAV 2025

Space Explanations

- We generalize the definition of Inflated Explanations as follows: A Space Explanation for class c is a logical formula $\varphi(\mathbf{x})$ such that

$$\forall(x \in \mathbb{F}) : \varphi(\mathbf{x}) \Rightarrow \kappa(\mathbf{x}) = c.$$

- Thus $\varphi(\mathbf{x})$ represents a sufficient condition for samples $\mathbf{x}$ to be classified as c.
- For φ we usually consider conjunctions of linear constraints, e.g.

$$\varphi(\mathbf{x}) = (x_1 + 3.4x_3 + x_5 \leq 9) \wedge (x_2 - 0.5x_5 \geq 3.4)$$

or

$$\varphi(\mathbf{x}) = \bigwedge_{j \in \mathcal{F}} (x_j \in \mathbb{E}_j) \qquad \text{or} \qquad \varphi(\mathbf{x}) = \bigwedge_{j \in \mathcal{A}} (x_j = v_j)$$

for intervals $\mathbb{E}_j$ respectively a subset of features $\mathcal{A} \subset \mathcal{F}$ – hence, Inflated Explanations and Abductive Explanations are a special case of Space Explanations.

- The formula φ defines a subset of the feature space in which it holds - the Impact Space.

Space Explanations

- Space Explanations are computed using advanced concepts from formal logical reasoning.
- Specifically, we need a systematic way to obtain a *general* formula (within the context of LRA) which serves as a sufficient condition for class c.
- This is given by the framework of Craig Interpolation and Unsatisfiable Cores.

Craig Interpolation

Theorem:
If $\varphi \Rightarrow \psi$, and φ and ψ have at least one variable in common, then there is a formula ρ (the Interpolant) which only uses logical symbols present in both φ and ψ, and

$$\varphi \Rightarrow \rho \qquad \text{and} \qquad \rho \Rightarrow \psi.$$

Example: In propositional logic, let

$$\varphi = \neg(P \wedge Q) \rightarrow (\neg R \wedge Q) \qquad \psi = (S \rightarrow P) \vee (S \rightarrow \neg R).$$

Then $\varphi \Rightarrow \psi$ (this can be seen by observing $\varphi \equiv (P \vee \neg R) \wedge Q$). We can use this to derive the interpolant $\rho = (P \vee \neg R)$, now

$$\varphi \Rightarrow \rho \qquad \text{and} \qquad \rho \Rightarrow \psi.$$

Acknowledgment: Example taken from Wikipedia, *Craig Interpolation*

Unsatisfiable Cores

Definition

Given a set of constraints F (a formula), an ***unsatisfiable core** is a subset $C \subseteq F$ such that:

- C is unsatisfiable, and
- every proper subset of C is satisfiable (i.e., C is **minimal**).

Intuition

A core pinpoints the minimal conflicting requirements.

How to obtain UCs

- Modern SAT/SMT solvers can return a (minimal or small) unsat core.
- Techniques: deletion filtering, resolution-based core extraction, hitting set dualization, MUS enumeration.

Example

Consider the constraint set F:

$$\begin{aligned} c_1 &: \quad x \geq 5 \\ c_2 &: \quad x \leq 3 \\ c_3 &: \quad y > 0 \\ c_4 &: \quad x + y \geq 7 \end{aligned}$$

F is unsatisfiable because $x \geq 5$ and $x \leq 3$ cannot both hold.

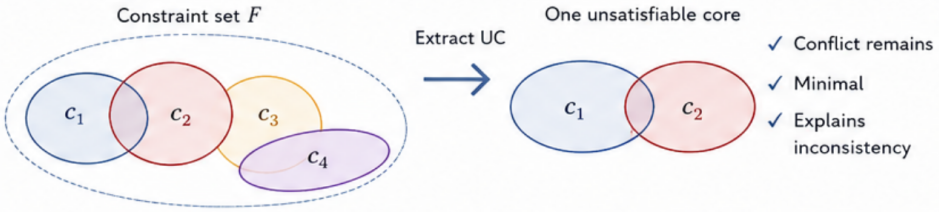
An unsatisfiable core of F is:

$$C = \{c_1, c_2\}$$

- C is unsatisfiable ($x \geq 3$ and $x \leq 3$ conflict)
- Any proper subset is satisfiable: $\{c_1\}$ ✓, $\{c_2\}$ ✓

Note: There can be multiple UCs.
Another UC here is $C' = \{c_1, c_4, c_3\}$ (since $x \geq 5, y > 0, x + y \geq 7$ is also inconsistent).

Visual view



Why UCs matter

Debugging & diagnosis
Identify the minimal source of inconsistency in requirements, knowledge bases, or configs.

Explainability (XAI)
Highlight the smallest set of inputs/rules that cause a contradiction or a decision.

Verification
Localize conflicting constraints in large verification problems.

Optimization & fixing
Suggest minimal changes to restore satisfiability (e.g., hitting set duals).

Key takeaway

Unsatisfiable cores are minimal explanations of inconsistency. They help explain, debug, and fix complex systems efficiently.

Image: ChatGPT Does anybody spot the hallucination?

Space Explanations

- For computing Space Explanations, we encode our task in the following way:
  - ➔ We encode the neural network and the domains of all input features as formula ψ_M and ψ_D, respectively.
  - ➔ Likewise, we encode the fact that a sample is classified as class c with a formula ψ_c.
  - ➔ The formula $\psi := \psi_M \wedge \psi_D \wedge \neg\psi_c$ means that a point $\mathbf{x}$ is *not* classified as class c.
- Clearly, the formula ψ is satisfiable (apart from pathological cases): there is a possible assignment for the input values $\mathbf{x}$ which leads to a class different from c.
- However, if we choose a specific sample $\mathbf{s}$ which *is* classified as class c, and let $\varphi_{\mathbf{s}} = \bigwedge_{i \in \mathcal{F}} (x_i = s_i)$, the formula

$$\varphi_{\mathbf{s}} \wedge \psi$$

is *unsatisfiable*.

Space Explanations

- Assume we aim at explaining why a specific sample **s** is classified is class c.
- The key to unlocking the power of Space Explanations lies in the unsatisfiable formula $\varphi_{\mathbf{s}} \wedge \psi$!
- This formula enables using Craig Interpolation. The system searches for an interpolant I such that $\varphi_{\mathbf{s}} \Rightarrow I$ and $I \wedge \psi$ is unsatisfiable.
- I now encodes a constraint on the input features (Q: why only on the input features and not on the rest of the NN?) which is *weaker* than the original constraint $\varphi_{\mathbf{s}} = \bigwedge_{i \in \mathcal{F}} (x_i = s_i)$, but still enforces that all samples described by (the Impact Space of) I are classified as class c.
- All formulas and constraints are expressed in Linear Real Algebra (LRA).
- Such search algorithms are an active field of research, we have implemented Space Explanations within the OpenSMT framework.

https://verify.inf.usi.ch/opensmt

Space Explanations

- Here is a high-level summary:
  - ➔ Start with the constraint $\varphi_{\mathbf{s}} = \bigwedge_{i \in \mathcal{F}} (x_i = s_i)$ which fixes all input features according to sample $\mathbf{s}$: This is the smallest (and least interesting) Space Explanation for this sample.
  - ➔ Relax the constraints on the input features while keeping rest of the formula valid (classification cannot change).
  - ➔ This produces increasingly more general Space Explanations (i.e. with greater Impact Space).
- The Space Explanation corresponds to a polytope in the feature space.
- If the Space Explanation is in CNF, the polytope is convex (makes many processing steps easier).
- Computing Space Explanations is still relatively slow, but recently we have been making great progress!

Space Explanations

- Space Explanations can take the form of rather complex formulas, corresponding to polytopes with a large number of faces.
- Ongoing work on interpreting Space Explanations[12]:

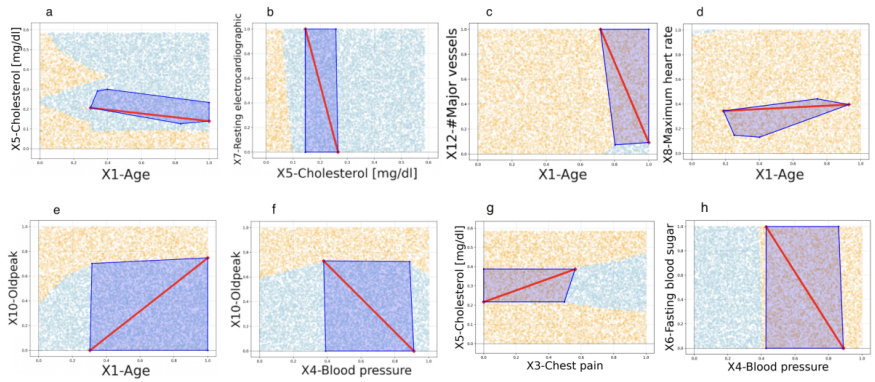


Fig. 3: Maximum diameters (red) of space explanations (purple)

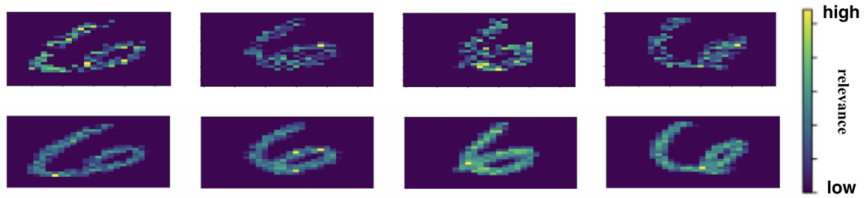


Fig. 4: Visualization of minimum norm points within space explanations

[12] Leopardi et al: *Interpreting Logical Explanations of Classifying Neural Networks*. Proc. ESANN 2025

idsia

Conclusion

Conclusion

- XAI has been investigated for quite some time.
- The "classical" methods usually offer local approximations of the NN behavior, but yield no formal or logical guarantees.
- Several novel methods provide different views on what an "Explanation" actually is, and what it entails.
- Formal Reasoning provides a principled way to obtain versatile explanations of the behavior of NNs, with two major features:
  - ➔ Precise formulas and logical guarantees for explanations
  - ➔ A powerful search technique to find these explanations.